# 2-24-1 Optimization and search heuristics

Baptiste Louf, Yann Ramusat

February 2, 2017

For the purpose of the project we have implemented most of the algorithms seen during the first part of the course. In addition to the *RLS* and *(1+1)EA* that are already part of the bootstrap project we propose the *($\mu$+$\lambda$)EA*, the *($\mu$,$\lambda$)EA*, the *simulated annealing* and the *1+($\lambda$,$\lambda$)GA*.

We also use an heuristic to identify useless object and delete them. You can find more information for this processing in INSERT REF.

We will discuss about the impact of the preprocessing, the optimal parameters $\lambda$ and $\mu$ for each algorithm and we will compare the differents heuristics over different kind of instances.

TODO PRECISE CONFIG OF THE PC.

## Contents

# 1 Preprocessing

We begin with a comparison of the effectiveness of the *RLS* with and without the preprocessing. This will give an insight of the utility of this method. We consider the dataset **fnl4461 n4460 bounded-strongly-corr 01**.

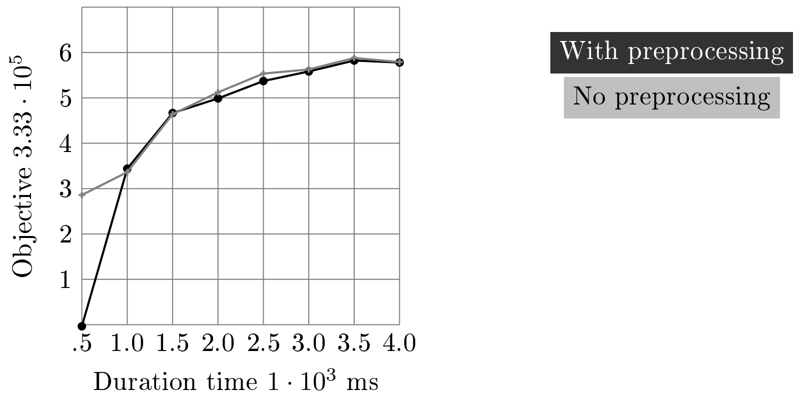


Figure 1: Effectiveness with and without preprocessing.

This suggests the preprocessing is not so usefull. In the following tests we will assume the preprocessing is desactivated.

# 2 Finding optimal parameters

We now search for optimal values for the parameters given to the *Evolutionary Algorithms.*

## 2.1 Parameters for the $(\mu+\lambda)$EA

We have computed results over the same dataset as before but with a time limit fixed to 2 seconds.

In the following table we give the objective value $(\cdot 10^3)$ depending on $\lambda$ (columns) and $\mu$ (rows).

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 57.319 | 75.313 | 82.962 | 79.675 | 90.212 | 79.101 | 83.769 | 83.558 |
| 2 | 0.94 | 28.786 | 47.334 | 55.418 | 52.986 | 42.394 | 57.919 | 58.145 |
| 3 | -47.932 | -8.061 | -0.737 | 13.734 | 34.343 | 24.937 | 25.238 | 30.463 |

We do not represent greater values of $\mu$ because they are worth.

We observe that definitively, the best value for $\mu$ is 1 and that $\lambda$ can be taken around 5 or 6.

## 2.2 Parameters for the $(\mu,\lambda)$EA

We still use the same dataset and the same time limit.

| | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|
| 1 | 83.061 | 76.424 | 79.828 | 90.550 | 84.161 | 89.578 | 86.279 | 80.184 |
| 2 | 57.759 | 67.847 | 67.239 | 64.593 | 60.688 | 64.267 | 72.076 | 67.081 |
| 3 | 15.680 | 40.752 | 48.155 | 58.068 | 57.982 | 59.098 | 59.939 | 55.429 |

For the two algorithm the impact of the values are similar (except when $\mu$ and $\lambda$ are equals to 1).

We conclude optimal parameters are 1 for $\mu$ and around 10 for $\lambda$.

# 3 Comparisons between algorithms over different datasets

## 3.1 Optimum reached

| a280 | strongly corr | uncorr/sim weights | uncorr/diff weights |
|---|---|---|---|
| RLS | 15051.8019370 | 104363.6465698 | 411714.7895653 |
| $(\mu+\lambda)$EA | 15964.2232060 | 104364.48085275 | 406518.83992591 |
| Sim. annealing | 14574.5134533 | 104364.02100352 | 411714.7895653 |
| $(\mu\lambda)$EA | 12780.4646832 | 103668.17130203 | 405539.75820266 |
| $1+(\lambda\lambda)$GA | 15601.751609 | 104196.08949973 | 329082.5109551 |
| Aléatoire | 14430.5244282 | 103900.45630387 | 382653.85033683 |

| fnl4461 | strongly corr | uncorr/sim weights | uncorr/diff weights |
|---|---|---|---|
| RLS | 200876.0190944 | 1478962.470699 | 6256257.38687 |
| $(\mu+\lambda)$EA | 216245.328453 | 1478185.8014 | 4886149.62120 |
| Sim. annealing | 199927.6417853 | 1478962.454439 | 6259185.35060 |
| $(\mu\lambda)$EA | 108355.7323370 | 1451984.570510 | 5259059.98241 |
| $1+(\lambda\lambda)$GA | 188920.0234235 | 1408012.654966 | 2846992.70523 |
| Aléatoire | 114309.8020697 | 1436582.160823 | 3962808.83123 |

## 3.2 Convergence

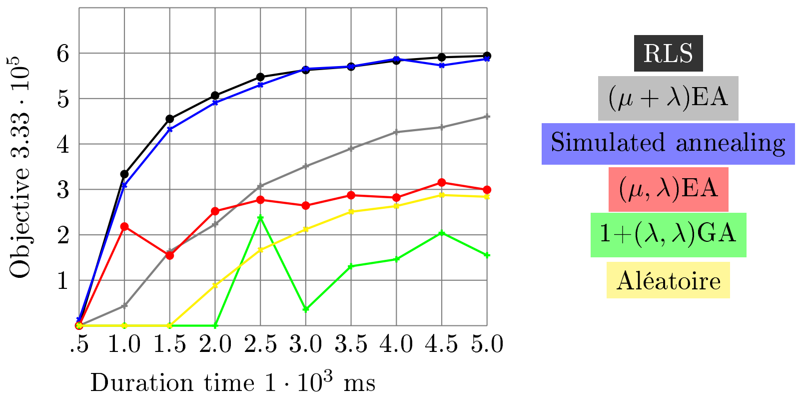


Figure 2: Convergences of different algorithms (basic dataset).

# 4 Outline of the algorithm(s) and choices made

insert big outline

## 4.1 Finding the tour

- not recomputed - no mutation of the tour + explanation

## 4.2 Preprocessing the instance

- explanation - why we dropped it : not so efficient + overfitting

## 4.3 Search Heuristics

### 4.3.1 Random Local Search

### 4.3.2 (1+1) EA

### 4.3.3 ($\mu$+$\lambda$) EA

-fixing parameters (for each instance ?)

### 4.3.4 ($\mu$,$\lambda$) EA

-fixing parameters (for each instance ?)

### 4.3.5 1+($\lambda$,$\lambda$) GA

-take big parameters

### 4.3.6 Considering several mutations

-consider which mutations ? gotta fix parameters

# 5 Experimental results

- explicit the instances we're working on - methodology : 10 minutes max + 10000 without change, algo launched 5 times - compare to competition (be honest), and to gecco paper - some convergence speed graphs to illustrate - un tableau qui regroupe tout en résultat total